REPORT DOCUMENTATION PAGE

Form Approved
OMB NO. 0704-0188

1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	January 15, 2006	Final Report (Sep. 1 2001 to Aug. 31 2005)

4. TITLE AND SUBTITLE

Supervisory Control of Networked Control Systems

5. FUNDING NUMBERS

DAAD19-01-1-0743

6. AUTHOR(S)

Michael Lemmon

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Dept. of Electrical Engineering, University of Notre Dame
Notre Dame, IN 46556

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

U. S. Army Research Office
P.O. Box 12211
Research Triangle Park, NC 27709-2211

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

41918.1-CI

11. SUPPLEMENTARY NOTES

The views, opinions and/or findings contained in this report are those of the author(s) and should not be construed as an official Department of the Army position, policy or decision, unless so designated by other documentation.

12 a. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for public release; distribution unlimited.

12 b. DISTRIBUTION CODE

.

13. ABSTRACT (Maximum 200 words)

This project's objective was to develop methods for the reliable and robust control of dynamical systems over wireless communication networks. Specific accomplishments of this project were the identification of fundamental bounds on the optimal control system performance achievable with bit-rate limited feedback, liveness-enforcing supervisory control of concurrent systems modelled by Petri nets, methods ensuring cohesive swarming of multi-agent systems under consensus filtering, adaptive real-time scheduling methods for feedback control, and the development of a wireless autonomous mobile robot testbed.

14. SUBJECT TERMS

Networked Control Systems – performance – stability – multi-agent swarms – consensus filtering

15. NUMBER OF PAGES

24

16. PRICE CODE

17. SECURITY CLASSIFICATION OR REPORT	18. SECURITY CLASSIFICATION ON THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	**UNCLASSIFIED**	**UNCLASSIFIED**	**UL**

TABLE OF CONTENTS

Supervisory Control of Networked Control Systems

Michael Lemmon
Dept. of Electrical Engineering
University of Notre Dame
Notre Dame, IN 46556, USA
574-631-8309 (fax: 574-631- 4393)
lemmon@nd.edu

ARO GRANT Proposal Number: 41918-CI
FINAL TECHNICAL REPORT: January 15, 2006

Problem Statement:

A networked control system is a control system whose feedback path is realized over a computer communication network. Networking feedback paths in such a manner can greatly reduce the hardware complexity of a large control system. Networked systems are often easier to manage and reconfigure. Traditional examples of networked control systems include networking of PID loop in the control of large-scale process control facilities. More recent applications involve the coordinated control of multiple autonomous vehicles over ad hoc wireless networks.

In many networked control applications, the network is shared between many different users. These users can be other control loops or they can be data logging tasks. In applications using ad hoc wireless networks, the bandwidth available to a single control loop may be severely limited. Moreover, competition for network resources between different users may reduce the reliability with which a single feedback path can access network resources. Network bandwidth in such applications is difficult to predict ahead of time and the inevitable variations in such bandwidth can severely compromise the anticipated stability and performance of the closed loop system.

The purpose of this project was to study the impact that limited feedback information has on the stability and performance of networked control systems. This project was extremely interested in establishing fundamental bounds on the best performance achievable over feedback links with fixed bit-rates and randomly dropped data packets. It attempted to use these theoretical results to develop real-time scheduling protocols that would minimize the impact that firm deadlines have on control system performance. Finally, this project studied the coordinated control of multi-agent autonomous systems when there is limited communication between agents. The project sought to build a wireless robotic testbed to study issues associated with the real-life implementation of networked control over ad hoc wireless networks.

Summary of Results:

The primary results from this project will be found in the published conference and journal papers listed in the following section. These results are itemized below.

1. This project developed a method for designing liveness-enforcing supervisors of concurrent discrete-event systems. The synthesis method was based on a partial order analysis of Petri nets. The publications describing this work are [He02] [Lem02].
2. This project developed methods for analyzing the controlled composition of hybrid automata through the use of viability kernels [Shang02].
3. This project analyzed the impact that independent and identically distributed dropouts had on control system performance. Our initial efforts used a power spectral analysis of a linear scalar feedback system with dropouts to study performance [Ling02]. Subsequent work developed a dropout compensator that minimized the impact such dropouts had on closed loop performance [Ling03b] [Ling04].
4. This project analyzed the impact that dropouts generated by a Markov chain had on the performance of linear feedback control systems. Using results from jump linear systems theory, we characterized the closed loop system performance [Ling03a] as a function of the dropout process' Markov chain. We used these results to determine a Markov chain that minimized the overall performance loss due to dropouts under a fixed average dropout rate.
5. Results from [Ling03a] were used to develop a firm real-time scheduling constraint. We used the optimal dropout Markov chain as a specification on the desired dropout patterns that a firm real-time scheduler should attempt to enforce. Several heuristic algorithms were proposed for realizing this constraint. Simulation results [Hu03] [Liu05] [Liu06] demonstrated that the MC-constraint could indeed be realized in a way that approaches the theoretical bounds derived in [Ling03a].
6. The feedback information in a networked control system is quantized due to the digital nature of network communication. This project focused on developing bit allocation strategies that maximize the overall closed loop performance. In a quantized feedback system, we have a fixed number, Q, bits that are used to encode the state vector. Bit assignment protocols determine how many of these Q bits should be allocated to a particular component of the state vector. In this part of the project we studied optimal bit assignments that minimized the error of dynamically quantized feedback strategies.

   The impact that dynamic bit assignment has on the asymptotic stability of noise free systems was examined in [Ling04b] [Ling05]. The performance of scalar linear systems with dynamically quantized feedback was studied in [Lem04]. The stability of quantized linear systems with bounded noise under dynamic bit assignment was studied in [Ling04a]. A characterization of the optimal performance (minimum quantization error) achieved under dynamic quantization for second order noise-free linear systems was presented in [Ling05a]. We've recently [Lem06] extended the work in [Ling05a] to characterize the optimal performance achieved in multivariable linear systems with bounded noise. The work in [Lem06] essentially generated what may be called *performance-rate* functions for dynamically quantized feedback

systems. A brief synopsis of that unpublished manuscript will be found in the technical appendix.

7. Recent results have examined the performance and stability of multi-agent robotic systems that communicate over ad hoc wireless networks. We [Sun05] proposed the use of periodic communication logics for the decentralized control of multi-agent systems. This work characterized the optimal periodic schedule of message transmissions that minimize average the mean square estimation error within the multi-agent system. We've recently [Lem06a] determined uniform ultimate bounds on multi-agent swarms that are moving under the guidance of a consensus filter. This work also added integral action to ensure asymptotic stability of the consensus filter. A brief synopsis of the unpublished manuscript [Lem06a] will be found in the technical appendix.
8. This project built a multi-robotic testbed consisting of 3 Koala robots [Lem06b]. The robots are controlled by MICA2 wireless processor modules. The robots communicate over the MICA2's radio. The network middleware implements a consensus filter that estimates the geometric center of the robotic group. This estimate is then used to guide the motion of the entire group. A more detailed description of the testbed [Lem06b] will be found in the technical appendix.

Publications:

Refereed Journal Papers

1. [He02] K.X. He and M.D. Lemmon (2002), *Liveness-Enforcing Supervision of Bounded Ordinary Petri Nets using Partial Order Methods", IEEE Transactions on Automatic Control*, July 2002, Volume 47, pages 1042-1055.
2. [Ling05] Q. Ling and M.D. Lemmon (2005), *Stability of Quantized Control Systems under Dynamic Bit Assignment,* IEEE Transactions on Automatic Control, volume 50, June 2005.
3. [Ling04] Q. Ling and M.D. Lemmon (2004), *Power Spectral Analysis of Networked Control Systems with Data Dropouts*, IEEE Transactions on Automatic Control, Volume 49(6), pages 955-959, June 2004.
4. [Liu06] D. Liu, X. Hu, M.D. Lemmon, and Q. Ling (2006), *Firm Real-Time System Scheduling Based on a Novel QoS Constraint*, IEEE Transactions on Computers, March, 2006.

Referred Conference Papers:

1. [Hu03] Sharon Hu, Donglin Liu, , Michael D. Lemmon, and Qiang Ling (2003), Firm Real-Time System Scheduling Based on a Novel QoS Constraint, Real Time Systems Symposium (RTSS'03), Cancun Mexico, November 2003.
2. [Lem02] M.D. Lemmon and K.X. He (2002), *Liveness enforcing monitors for safe and controllable Petri nets,* 2002 IEEE Conference on Decision and Control, Las Vegas, Nevada, December 2002.
3. [Lem04] M.D. Lemmon and Q. Ling (2004), *Control system performance under dynamic quantization: the scalar case*, Conference on Decision and Control (CDC'04), Bahamas, Dec. 2004
4. [Ling03] Q. Ling and M.D. Lemmon (2003), *Optimal Dropout Compensation in Networked Control Systems*, IEEE Conference on Decision and Control (CDC03), Hawaii, December 2003.
5. [Ling03a] Q. Ling and M.D. Lemmon (2003), *Soft real-time scheduling of networked control systems with dropouts governed by a Markov chain*, American Control Conference, Denver, Colorado, June 2003.
6. [Ling03b] Q. Ling and M.D. Lemmon (2002), *Robust performance of soft real-time networked control systems with data dropouts*, 2002 IEEE conference on decision and control, Las Vegas, Nevada, December 2002
7. [Ling04a] Q. Ling and M.D. Lemmon (2004), *Stability of quantized linear systems with bounded noise under dynamic bit assignment,* Conference on Decision and Control (CDC'04), Bahamas, Dec. 2004
8. [Ling04b] Q. Ling and M.D. Lemmon (2004), *Stability of Quantized Control Systems under Dynamic Bit Assignment,* American Control Conference (ACC'04), Boston, MA, Dec. 2004
9. [Ling05a] Q. Ling and M.D. Lemmon (2005) , Optimal dynamic bit assignment in noise-free quantized linear control systems, I*EEE Conference on Decision and Control*, Seville Spain, Dec. 2005.

10. [Liu05] D. Liu, X. Hu, M.D. Lemmon, and Q. Ling (2005), Scheduling Tasks with Markov-Chain Constraints, 17th Euromicro Conference on Real-time Systems, July 2005.
11. [Shang02] Y. Shang and M.D. Lemmon (2002), *The controlled composition of hybrid automata through the use of inner viability kernels*, MTNS 2002, Univ. of Notre Dame, August 2002.
12. [Sun05] Y. Sun and M.D. Lemmon (2005), Periodic Communication Logics for the Decentralized Control of Multi-agent Systems, IEEE Conference on Control Applications, Toronto, Canada, August 2005.

Manuscripts in Preparation:
1. [Lem06] M.D. Lemmon and R. Sun (2006), Optimal Dynamic Bit Assignment for Dynamically Quantized Feedback Systems, to be submitted to IEEE Conference on Decision and Control 2006.
2. [Lem06a] M.D. Lemmon and Y. Sun (2006), Cohesive Swarming under Consensus, to be submitted to IEEE Conference on Decision and Control 2006.
3. [Lem06b] M.D. Lemmon and Y. Sun, Cohesive Swarming under Consensus: robotic testbed, in preparation, 2006.

Invited Presentations:
1. Fundamental Limitations of Networked Control due to Dropouts and Quantization, Old Dominion University, Virginia, October 2003.
2. Fundamental Limitations of Networked Control due to Dropouts and Quantization, Queens University, Canada, October 2003.
3. Stability and Performance of Feedback Control Systems over Wireless Sensor-Actuator Networks, Purdue University, April 2004.
4. Detection and Control of Combined Sewer-Overflow Events using Embedded Sensor Network Technology, Purdue University, November 2004.
5. Control of Environmental Processes over Embedded Sensor-Actuator Networks, Johnson Controls Inc., Milwaukee, April 11, 2005.
6. Detection and Control of Combined Sewer Overflow Events using Embedded Sensor Network Technology, June 2005, Notre-Dame/Johnson Control/City of South Bend ideation session.
7. Sensor Networks at Notre Dame, Hewlett-Packard Research Center, Palo Alto California, August 2005.
8. Embedded Sensor-Actuator Network Research at Notre Dame, Argonne National Laboratories, October, 2005
9. Embedded Sensor-Actuator Network Research at Notre Dame, General Electric (remote diagnostics division), November 2005.
10. Monitoring and Control of Environmental Processes over Embedded Sensor-Actuator Networks, Crane Naval Base, Evansville Indiana, May 18, 2005.

Scientific Personnel:

1. Michael Lemmon, Professor, Dept. of Electrical Engineering, University of Notre Dame
2. Kevin X. He, Ph.D., Morgan, Lewis and Bockius LLP, New York, NY (Ph.D. completed under this grant in 2002).
3. Qiang Ling, Ph.D., Seagate Technologies, Pittsburgh, PA (Ph.D. completed under this grant in 2005)
4. Yashan Sun, Dept. of Electrical Engineering, University of Notre Dame, Ph.D. student. (Ph.D. expected spring 2007)
5. Ying Shang, Dept. of Electrical Engineer, University of Notre Dame, Ph.D. student, (Ph.D. degree expected spring 2006)
6. Donglin Liu, Dept. of Computer Science and Engineering, University of Notre Dame, M.S. student (degree completed Summer 2005)

Report of Inventions: None

Appendix A - Performance-Rate Functions for Quantized Feedback Systems

Consider a quantized feedback system shown in figure 1A The discrete-time system has a state *x[k]* that satisfies the state equations,

$$x[k+1] = Ax[k] + Bu[k] + w[k]$$

for $k=0,\ldots,\infty$. The signal, *w[k]*, is an exogenous disturbance such that $|w[k]|_\infty \le M$, where *M* is a finite constant. The signal *u[k]* is the feedback control law of the form

$$u[k] = Fx^q[k]$$

where $x^q[k]$ is a *quantized* approximation of the state at time *k*. We assume that *(A,B)* is controllable with *A* being diagonalizable with unstable eigenvalues ($|\lambda_i|>1$) and *F* is a stabilizing state feedback gain matrix of appropriate dimensions.

Figure 1A

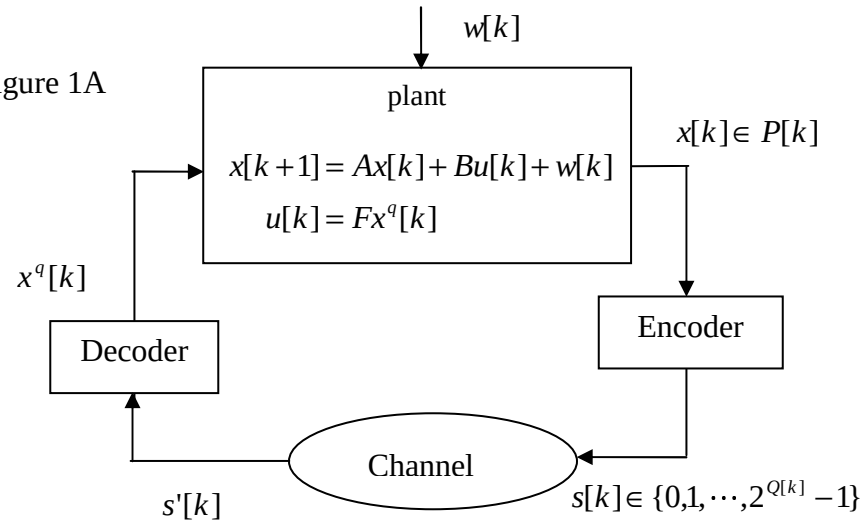


Figure 1A shows how the system state, *x[k]*, is quantized and transmitted over the feedback channel. In this figure, the system state is measured at time *k* by the *encoder* and that measurement is mapped onto a symbol *s[k]* that is drawn from the discrete set $\{0,1,\ldots,2^Q-1\}$. The symbol *s[k]* is therefore represented by *Q* bits. This symbol is transmitted across a lossless communication channel with a single step delay. The *decoder* receives a symbol *s'[k]=s[k-1]* that is a one-step delayed version of the transmitted symbol. The decoder uses *s'[k]* to construct the quantized approximation, $x^q[k]$, of the system state.

The quantization method used to construct $x^q[k]$ originates in the *uncertainty set evolution method* introduced in [Brockett]. This approach presumes that at the beginning of the *k*th time interval, the encoder and decoder agree that the state lies in the set

$$x[k] \in x^q[k] + U[k].$$

U[k] is a rectangular set of the form

$$U[k] = \prod_{i=1}^{n} \left[-L_i[k], L_i[k] \right]$$

where $x^q[k]$ is the quantized state and $L_i[k]$ is the half length of the ith side of the rectangle $U[k]$ at time k. We sometimes refer to $U[k]$ as the *uncertainty set*. The *quantization error* between the true state and quantized state is denoted as $e[k] = x[k]-x^q[k]$.

The encoder measures the system's current state $x[k]$ immediately after the start of the kth time interval. The encoder then uses this measurement to determine that

$$x[k] \in x^q_{s[k]} + U_{s[k]}$$

where $x^q_{s[k]}$ is the center of a smaller subset and

$$U_{s[k]} = \prod_{i=1}^{n} \left[-\frac{L_i[k]}{2^{b_i[k]}}, \frac{L_i[k]}{2^{b_i[k]}} \right]$$

$b_i[k]$ represents the number of bits used to quantize the ith component of the state vector at time k. The new center and smaller uncertainty set are indexed by the symbol $s[k]$ which is transmitted across the communication link with a one-step delay.

The decoder receives the symbol, $s[k]$ at time $k+1$. As soon as it receives the symbol, it knows that the system state at time k lies in the new uncertainty set $x^q_{s[k]} + U_{s[k]}$. However, time has now marched ahead from k to $k+1$, so the decoder must propagate the uncertainty set through the state dynamics to determine the state at time $k+1$. This is done by the following equations

$$x[k] \in x^q[k+1] + U[k+1]$$

$$U[k+1] = \prod_{i=1}^{n} \left[-L_i[k+1], L_i[k+1] \right]$$

$$x^q[k+1] = Ax^q_{s[k]} + BFx^q_{s[k]}$$

$$L_i[k+1] = \frac{\lambda_i}{2^{b_i[k]}} L_i[k] + M$$

Throughout this paper we impose a *constant bit rate* constraint which require $b_i[k] \geq 0$ and

$$\sum_{i=1}^{n} b_i[k] = Q$$

where Q is a fixed positive integer representing the number of bits used to encode the state.

This paper will determine bit assignments that are *optimal* in the sense of minimizing the worst-case summed quantization error over a finite horizon of length N. In other words, we seek to minimize

$$P = \sup_{x[0]} \sum_{k=1}^{N} \sum_{i=1}^{n} \left(x[k] - x^q[k] \right)^2$$

through our bit assignments, $b_i[k]$ for a fixed Q. Note that because we take the sup over all initial condition, there exists a worst case initial condition such that this squared quantization error is in fact equal to $L_i[k]$. So the optimization problem can be restated as

$$\begin{aligned} \min \quad & \sum_{k=1}^{N}\sum_{i=1}^{n} L_i^2[k] \\ \text{with respect to} \quad & b_i[k] \\ & b_i[k] \geq 0 \\ \text{subject to} \quad & Q = \sum_{i=1}^{n} b_i[k] \\ & L_i[k+1] = \frac{\lambda_i}{2^{b_i[k]}} L_i[k] + M \end{aligned}$$

This problem will be solved using dynamic programming. The basic result is described below. Proofs have been omitted.

Proposition: If

$$Q > \sum_{i=1}^{n} \log_2 \lambda_i$$

$$\frac{Q}{n} = \frac{\bar{\Lambda}[0]}{\Lambda_i[0]}$$

for $i=1,\ldots,n$, then the bit assignment

$$b_i[k] = \frac{Q}{n} - \log_2\left(\frac{\bar{\Lambda}[k]}{\Lambda_i[k]}\right)$$

locally optimizes the bit assignment problem.

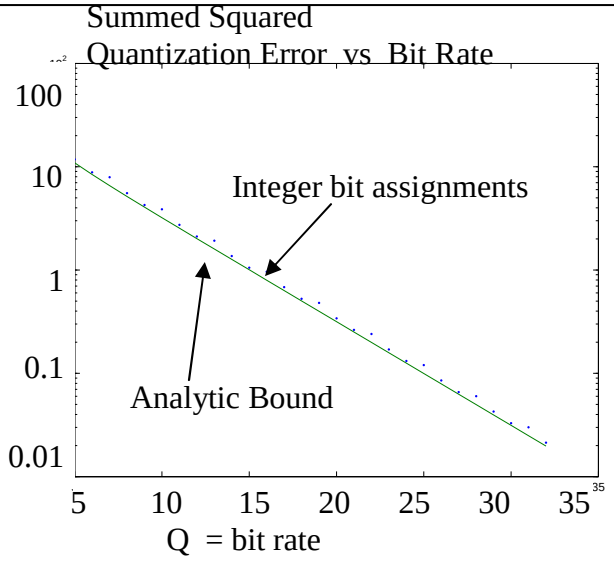
Remark: Note that the first constraint is the stabilizing bit rate constraint for unstable systems. The second constraint requires that the initial uncertainty set be *balanced*. Note that the bit assignment has an intuitive interpretation that requires us to equally distribute the Q available bits between all channels and then to adjust that "average" bit assignment to the ith side based on the balance between that side's uncertainty set and the geometric average of all sides of the uncertainty set. This means that optimal bit assignments seek to balance the uncertainty amongst all components of the state vector.

The preceding proposition may be proven using dynamic programming. The proof is interesting for it characterizes the optimal bit assignment as a feedback control law that is a function of the current uncertainty sets. Moreover, we can find an explicit closed form expression for the value function, which provides what we can think of as a *performance-rate function*. This is a function that plots the optimal achievable performance (minimum quantization error) as a function of the constant bit rate Q. For example, if we consider an *n-dimensional* linear system with noise amplitude M and take the horizon N to go to infinity, then the optimal performance as a function of Q is

$$P^* = nM^2\left(1+\frac{\bar{\lambda}}{2^{Q/n}(1-\rho)}\right)^2$$

where $\bar{\lambda}$ is the geometric average of the system's eigenvalues and $\rho = \frac{\bar{\lambda}}{2^{Q/n}}$. Figure 2A plots the predicted performance levels for a 3-d system and the associated simulation results. The figure shows close agreement between the predicted and simulated results. To our best knowledge, this is one of the first complete characterizations of the optimal performance attainable with multivariable quantized linear systems with noise.

Figure 2A – performance rate curve




```
for i=n:-1:1
   [tmp,imin]=min(Lambar./Lam);
   bc=Qm/i- log2(tmp);
   if i==1;
      b(imin)=Qm;
   else
      b(imin)=ceil(bc);
   end
   Qm=Qm-b(imin);
   Lam(imin)=.00001;
end
```

Appendix B - Cohesive Swarming under Consensus

Problem Statement: Consider a swarm of *N* dynamical agents that exchange information over an ad hoc communication network. Each agent is characterized by two types of states; its *physical state* representing the agent's position in the real world and its *consensus state* representing the agent's esetimate of the swarm's geometric center. The physical state of the *i*th agent at time *t* is denoted as the vector $x_i(t)$ in Euclidean *n*-space. The trajectory of the *i*th agent's physical state satisfies the ordinary differential equation

$$\dot{x}_i(t) = u_i(t) + \sum_{j \neq i} g(\| x_i - x_j \|)(x_i - x_j)$$

The vector u_i is an external input and $g:R \rightarrow R$ is a function representing long-range interactions between agents. We use the notation g_{ij} to denote $g(||x_i-x_j||)$.

The summation represents long range attraction and short-range repulsive interactions between agents. In particular, we assume this interaction may be decomposed into an attractive and repulsive component as

$$g(r) = \rho(r) - \alpha(r)$$

This paper restricts its attention to attraction and repulsion functions of the form

$$\rho(r) = \frac{\rho_0}{r^2}$$

$$\alpha(r) = \frac{\alpha_0}{r}$$

where ρ_0 and α_0 are positive constants.

The consensus state of the *i*th agent at time *t* is denoted as a vector $\hat{x}_i(t)$. The trajectory of the consensus state satisfies the consensus filter equations [Saber-Olfati/Shamma CDC2005],

$$\frac{d}{dt}\hat{x}_i = (x_0 - \hat{x}_i) + \sum_{j \neq i}^{n} A_{ij}\left(\hat{u}_j - \hat{x}_i\right) + \sum_{j \neq i}^{n} A_{ij}\left(\hat{x}_j - \hat{x}_i\right)$$

The vector x_0 is the *target* state that the swarm is trying to track. The coefficients A_{ij} are the elements of the matrix *I+Adj(G)* where *Adj(G)* is the adjacency matrix of a directed graph *G*. This directed graph models the communication network's connectivity within the swarm.

Figure 1B

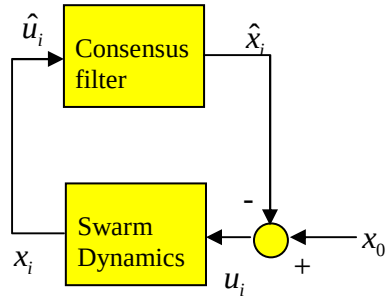


Figure 1B shows the entire swarm may be viewed as an interconnection of the swarm dynamics equation and the consensus filter.

The swarm dynamic's input from the *j*th agent to the consensus filter's *i*th agent is the position of the *j*th agent. In other words, $\hat{u}_j = x_j$. The consensus filter's input from the *j*th agent to the swarm dynamic's *i*th agent is

the estimate of the swarm center (consensus state) relative to the target. In other words, $u_j = x_0 - \hat{x}_j$. The consensus filter tries to estimate the center of the swarm and the swarm uses those estimates to guide the swarm toward the target. The overall dynamics of this system may therefore be written as

$$\frac{d}{dt}x_i = (x_0 - \hat{x}_i) + \sum_{j\neq i} g_{ij}(x_i - x_j)$$

$$\frac{d}{dt}\hat{x}_i = (x_0 - \hat{x}_i) + \sum_{j\neq i} A_{ij}(\hat{x}_j - \hat{x}_i) + \sum_j A_{ij}(x_j - \hat{x}_i)$$

We're interested in establishing whether or not the swarm is *cohesive* and achieves *consensus*. Let $\bar{x}(t) = \frac{1}{N}\sum_j x_j(t)$ denote the *swarm center* at time *t*. Define the *swarm error* and *consensus error* of the *ith* agent as

$$e_i(t) = x_i(t) - \bar{x}_i(t)$$

$$\hat{e}_i(t) = \hat{x}_i(t) - \bar{x}_i(t)$$

The swarm is cohesive if its swarm error is uniform ultimately bounded. The swarm achieves ε-consensus if the consensus error is less than a finite ε.

Error Equations: Since our analysis is concerned with the asymptotic behavior of the error vectors, it will be convenient to transform the original swarm dynamics and consensus filter into a set of coupled error equations. With some analysis it can be shown that these error equations are

$$\frac{d}{dt}e_i = \sum_{j\neq i} g_{ij}(e_i - e_j) + \frac{1}{N}\sum_j \left(\hat{e}_j - \hat{e}_i\right)$$

$$\frac{d}{dt}\hat{e}_i = -\hat{e}_i - \sum_j A_{ij}\hat{e}_i + \sum_{j\neq i} \bar{A}_{ij}\left(\hat{e}_j - \hat{e}_i\right) + \sum_j \left(A_{ij} - 1\right)e_j$$

where $\bar{A}_{ij} = A_{ij} + \frac{1}{N}$. The consensus error equation is sometimes more conveniently represented in matrix-vector form.

$$\frac{d}{dt}\hat{e} = A\hat{e} + Be$$

$$= \begin{bmatrix} -2(1+\Delta_1)I & \bar{A}_{12}I & \cdots & \bar{A}_{1N}I \\ \bar{A}_{21}I & -2(1+\Delta_2)I & \cdots & \bar{A}_{2N}I \\ \vdots & \vdots & \ddots & \vdots \\ \bar{A}_{N1}I & \bar{A}_{N2}I & \cdots & -2(1+\Delta_N)I \end{bmatrix} \begin{bmatrix} \hat{e}_1 \\ \hat{e}_2 \\ \vdots \\ \hat{e}_N \end{bmatrix} + \begin{bmatrix} 0 & (A_{12}-1)I & \cdots & (A_{1N}-1)I \\ (A_{21}-1)I & 0 & \cdots & (A_{2N}-1)I \\ \vdots & \vdots & \ddots & \vdots \\ (A_{N1}-1)I & (A_{N2}-1)I & \cdots & 0 \end{bmatrix} \begin{bmatrix} \hat{e}_1 \\ \hat{e}_2 \\ \vdots \\ \hat{e}_N \end{bmatrix}$$

where Δ_i denotes the out degree of the *i*th node.

Results: The main result establishes a bound on the level of consensus and the swarm size as a function of the swarm parameters. This is accomplished by studying the uniform ultimate boundedness (UUB) of the swarm dynamics and consensus filters.

The following lemma studies the directional derivative of a candidate Lyapunov function $V(e) = \frac{1}{2} e^T e$ and determines conditions under which it is guaranteed to be negative definite.

Lemma 1: Consider the swarm error system and let $V(e) = \frac{1}{2} e^T e$. If there exists a positive constant $\bar{\beta}$ such that

$$\sum_i \sum_j \| x_i - x_j \| \leq \bar{\beta} \| e \|$$

and if

$$\| e \| \geq \frac{N(N-1)\rho_0}{\bar{\beta}\alpha_0}$$

then $\dot{V}(e) \leq 0$.

A related *instability* result based on the above lemma characterizes the set of ||e|| for which $\dot{V}(e)$ is positive.

Lemma 2: Consider the swarm dynamics and let $V(e) = \frac{1}{2} e^T e$. If there exists $\underline{\beta} > 0$ such that

$$\underline{\beta} \| e \| \leq \sum_i \sum_j \| x_i - x_j \|$$

and if

$$\| e \| \leq \frac{N(N-1)\rho_0}{\underline{\beta}\alpha_0}$$

Then $\dot{V}(e) \geq 0$.

Remark: The above conditions in these lemmas are bounds on the average interagent distance. One natural question is to ask whether or not such bounds actually exist. This question can be answered as follows. Let $\| x \|_1 = \sum_i | x_i |$ denote the 1-normof the vector *x* and let $||x||_2$ denote the 2-norm. There always exist constants *c* and *C* such that

$$c \| x \|_2 \leq \| x \|_1 \leq C \| x \|_2$$

So now consider the swarm error vector *e* and note that

$$\| e \|_1 = \sum_i \frac{1}{N} \sum_j \| x_i - x_j \|_1$$

$$< \sum_i \frac{C}{N} \sum_j \| x_i - x_j \|_2$$

which implies there exists a constant $\underline{\beta}$ satisfying our constraint. Similar types of bounding arguments can be used to determine $\bar{\beta}$. So such bounds always exist.

Remark: The actual values for the bounds $\bar{\beta}$ and $\underline{\beta}$ may be obtained by solving the following optimization problem.

$$\begin{aligned} &\text{minimize} && J = \sum_i \sum_j \| e_i - e_j \|^2 \\ &\text{with respect to} && e_i \\ &\text{subject to} && \sum_e \| e_i \|_2^2 = E^2 \\ & && \sum_i e_i = 0 \end{aligned}$$

The optimal cost then becomes our bound.

Ultimate bounds can also be obtained for the consensus error. These bounds are given in the following lemma.

Lemma: Consider the consensus filter and let $V(\hat{e}) = \frac{1}{2}\hat{e}^T\hat{e}$. Let $\bar{\Delta}$ and $\underline{\Delta}$ denote the maximum and minimum out-degree of any node in the communication graph. If

$$\| \hat{e} \| > \frac{| N - \bar{\Delta} |}{1 + \bar{\Delta}} \| e |$$

then $\dot{V}(\hat{e}) \le 0$. If

$$\| \hat{e} \| \le \frac{| N - \bar{\Delta} |}{3(1 + \bar{\Delta})} \| e \|$$

Then $\dot{V}(\hat{e}) \ge 0$.

We can now establish the cohesion of the swarm under consensus by examining the regions identified in lemmas 1-3. This examination allows us to identify a compact region that is an attracting invariant set of the system. This result is stated in the following proposition.

Proposition: Consider the interconnected system and assume there exist $\underline{\beta}$ and $\bar{\beta}$ such that

$$\underline{\beta} \| e \| \le \sum_i \sum_j \| x_i - x_j \| \le \bar{\beta} \| e \|$$

Let

$$\Omega_s^- = \left\{ (e, \hat{e}) : \| e \| \ge \frac{N(N-1)\rho_0}{\bar{\beta}\alpha_0} \right\}$$

$$\Omega_s^+ = \left\{ (e, \hat{e}) : \| e \| < \frac{N(N-1)\rho_0}{\underline{\beta}\alpha_0} \right\}$$

$$\Omega_c^- = \left\{ (e, \hat{e}) : \| \hat{e} \| \ge \frac{| N - \bar{\Delta} |}{1 + \bar{\Delta}} \| e \| \right\}$$

$$\Omega_c^+ = \left\{ (e, \hat{e}) : \| \hat{e} \| < \frac{| N - \bar{\Delta} |}{3(1 + \bar{\Delta})} \| e \| \right\}$$

For any initial state $(e(0), \hat{e}(0))$, the set

$$\Omega = (\Omega_s^+)^c \cap (\Omega_s^-)^c \cap (\Omega_c^+)^c \cap (\Omega_c^-)^c$$

is an attracting invariant set.

Proof: The proof of this result follows immediately from the regions shown in figure 2B. This figure plots the 4 regions identified above and shows that the intersection of their complements is a compact set, Ω.

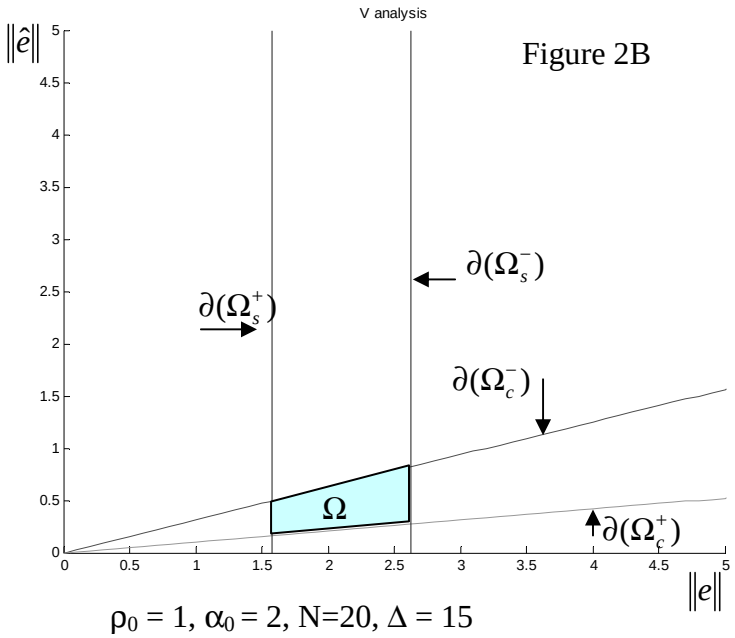


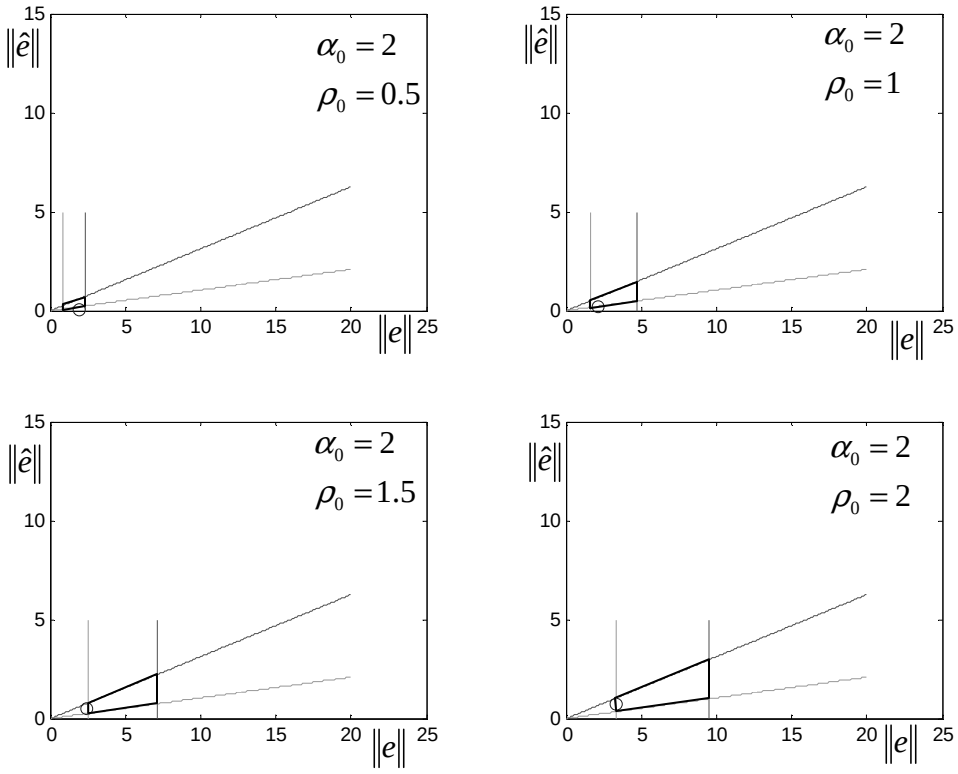
Figure 2B

$\rho_0 = 1$, $\alpha_0 = 2$, N=20, $\Delta = 15$

A Matlab simulation was written to simulate the system equations in equations. This simulation was performed with 20 agents in which the repulsion coefficient ρ_0 and the attraction coefficient, α_0 were both equal to one. The communication graph was specified at time 0 and that graph was kept static over the length of the run. This simulation's communication graph had a maximum connectivity of about $\overline{\Delta} = 15$. The swarm was attempting to intercept a target that started at (0,150) and moved with a constant velocity of (-10,-10). The swarm was initialized to be uniformly distributed over a rectangular region with side length 30 centered at the (15,15). The simulation was run for 100 time steps with a step size of T=0.02.

Figures 3B is similar to the plot shown in figure 2B. In this figure, however, not only do we plot Ω, but we show the final swarm and consensus errors achieved by the simulation. This final error vector is shown by the blue circle. The region Ω is marked by the dark black quadrilateral The four plots in figure 3B show these regions and simulation data assuming α_0=2 and with ρ_0 ranging from0.5 to 2.0. In viewing the plots, we want to see the experimental prediction lie within the set Ω. This happens in most cases with the simulation result usually resting at the far lefthand side of the set. Simulation results were outside of Ω when the simulation was run for too short a time, or else the integration step-size might have been too large to ensure simulation accuracy.

Figure 3B



Perfect Consensus: The consensus filter generates estimates of the swarm center which are then used by agents to guide the swarm to the target. The analytical bounds and simulation results presented above indicate that using the consensus filter, the best we can hope for is ε-consensus where the size of ε is given in preceding proposition. Obviously what we'd like to do is identify conditions under which we might drive ε to zero and thereby achieve perfect consensus.

One obvious way of achieving perfect consensus is through the introduction of integral action in the consensus filter equation. The state equations for the consensus filter with integral action are shown below,

$$\frac{d}{dt}\hat{e} = A\hat{e} + Be + KIz$$

$$\frac{d}{dt} = A\hat{e}$$

where *z* is the integrated error, *K* is an integrator gain and *I* is an Nn by Nn identity matrix.

To see how integral action achieves perfect consensus, let's first consider vectors $\hat{e}_{ss}$ and z_{ss} such that

$$0 = Be + KIz_{ss}$$

$$0 = A\hat{e}_{ss}$$

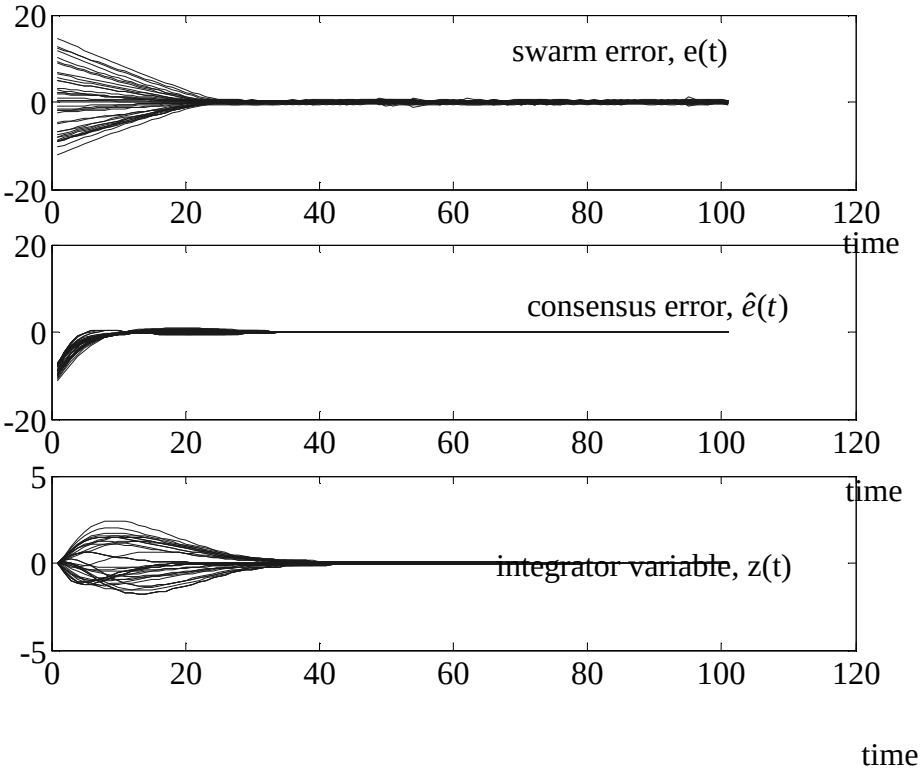
The augmented system equations may now be rewritten in matrix form as

$$\frac{d}{dt}\begin{bmatrix}\hat{e}\\ z\end{bmatrix}=\begin{bmatrix}A & KI\\ A & 0\end{bmatrix}\begin{bmatrix}\hat{e}\\ z\end{bmatrix}+\begin{bmatrix}B\\ 0\end{bmatrix}e$$

In the preceding equations, it should be apparent that $\hat{e}_{ss}$ and z_{ss} are equilibrium points of the unforced system (i.e. *e=0*). Inserting $\hat{e}_{ss}$ and z_{ss} into the above equation we see that $z_{ss}=\frac{1}{K}Be$ and $0=A\hat{e}_{ss}$. From our earlier work, we know that A is of full rank which means that $\hat{e}_{ss}=0$. Since this is the steady stte consensus error, we see the addition of integral action should enable perfect consensus.

A Matlab script was written to simulate swarming under consensus with integral action. In this particular simulation, we set K = 20 with α_0=1, ρ_0=2, N=20, and Δ=15. Figure 4B plots the swarm position error, the consensus error, and the integrator vector *z* as a function of time. The plots show that the consensus error clearly converges to zero for all agents.

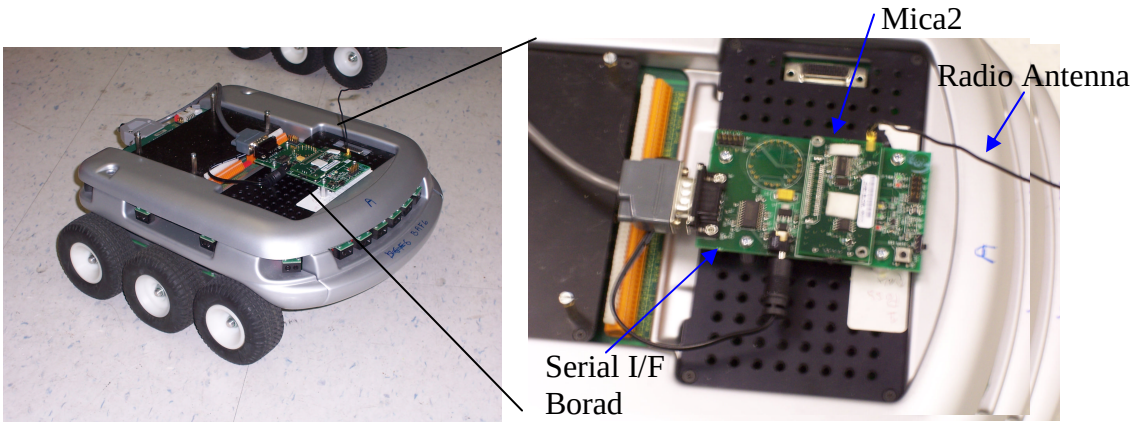
Figure 4B



Appendix C – Multi-agent Robotic Testbed

This paper provides preliminary documentation of a wireless autonomous robotic testbed. The system consists of 3 Koala (K-team Inc.) robots that are controlled by the MICA2 (Crossbow Inc.) wireless processor module. This testbed is being used to experimental test the swarming under consensus algorithms being developed by this project.

MICA-KoalaBot Hardware: The Koala robot is an autonomous wheeled vehicle that has 16 infrared (IR) proximity sensors around its perimeter. The robot's position is determined from encoders on the wheels. Basic motion commands and sensor commands are issued as ASCII strings to the robot over a RS-232 serial port. The Koala robot was augmented with a MICA2 processor. The MICA2 processor module is an embedded sensor node consisting of an 8-bit microprocessor and a Chipcon 1000 embedded radio. The MICA2 was connected to the Koala robot's serial port using Crossbow's MIB500 serial interface board. The MICA-KoalaBot is shown below.



Commands are issued to the robot over its serial I/F. The basic motor and sensor commands for this robot are listed below.

- **D,x,y**-set speed of right and left wheels to **x** and **y**, respectively.
- **E**-get wheel speed
- **G,x,y**-set wheel encoder counters (right/left)
- **H**-get wheel encoder counter values.
- **N**-get proximity sensor measurements

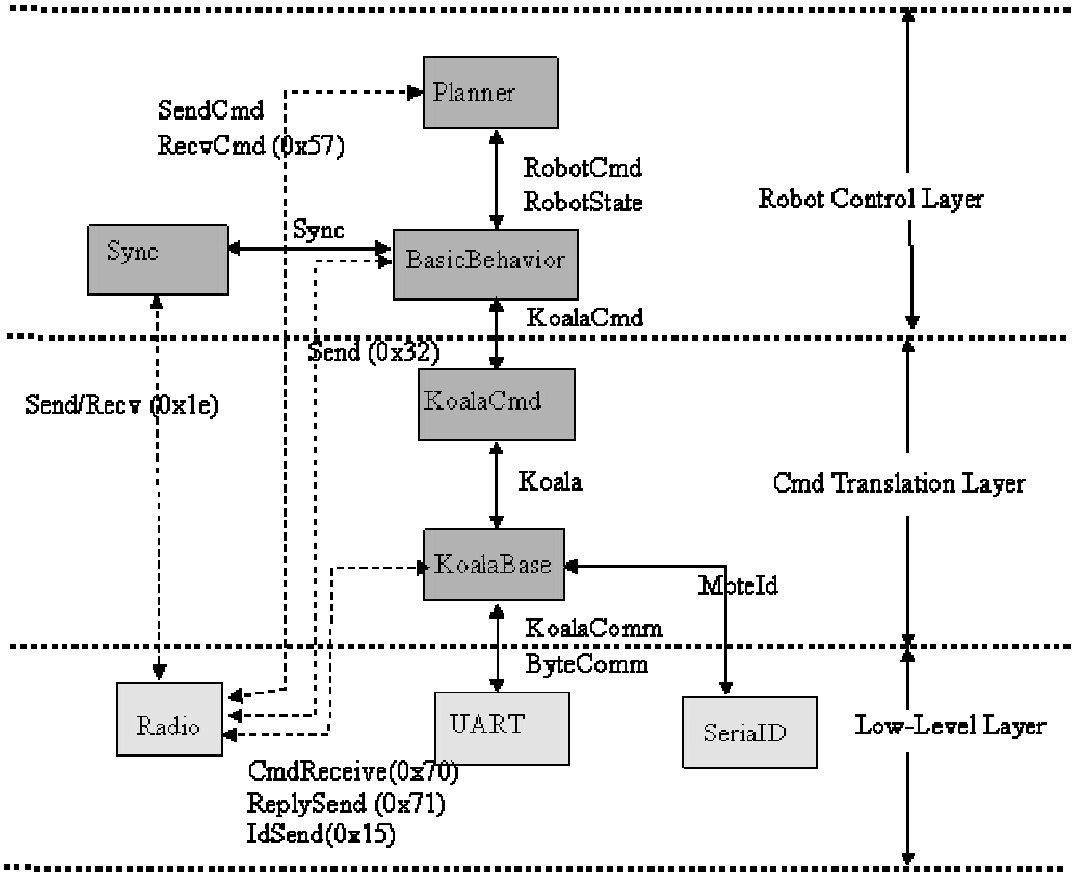
After receiving the command, the robot responds with an ASCII string that contains acknowledgement of the commands receipt and any data. For instance, if the MICA2 issues the command **N,** it is requesting the sensor data. A reply from the robot might take the form **n,21,13,14,15,16,20,255,233,250,120,34,23,24,25,45,50** which is the echoed lower case command letter followed by the 16 sensor measurements using commas as token separators.

MICA-KoalaBot Middleware: The MICA-KoalaBot is programmed using TinyOS. TinyOS is a object-oriented operating system developed at U.C. Berkeley to support the development of embedded sensor networks. TinyOS programs consist of software objects (components) that communicate with each other through well-defined signal interfaces. All computation within the system is in response to some received signal that may be

generated from the external environment (such as a sensor signal), from the processor hardware (i.e. a clock or radio), or from other TinyOS components. The MICA-KoalaBot software can therefore be viewed as a set of interconnected TinyOS components that encapsulate various tasks ranging from low-level control of the UART connection between the MICA2 and Koala to high-level updating of the vehicle's physical state. The major TinyOS components developed for the robot were

- **Planner** - high-level planning of basic robot commands
- **BasicBehavior** - implementation of raw robot commands
- **KoalaCmd** - translation of raw robot commands to ASCII string
- **KoalaBase** - sends and catches ASCII strings from robot
- **Sync** - clock synchronization
- **Low Level** - low level components controlling MICA2 hardware

These components are arranged in three layers as shown below. The highest layer control robot command functions, the second layer translates the commands into ASCII strings that are sent to the UART. The lowest level consists of the basic TinyOS components that control the MICA2 hardware.



Swarming under Consensus: The "swarming under consensus" algorithms have been implemented in the KoalaBot middleware. Swarming dynamics have been implemented as an obstacle avoidance strategy in the BasicBehavior component. These dynamics simply have the robot stop when it detects an obstacle in front of it. The consensus filter has been implemented in the PathPlanner component.

The consensus filter estimates the swarm's center. One important issue with regard to the implementation concerns the amount of network traffic. In order to reduce network traffic, a robot broadcasts its consensus state and true physical state only when it decides to change its direction. In other words, this platform implements an asynchronous version of the traditional consensus filter. In order to assure that all nodes have consistent estimates of the swarm's center, we use a fine-grained clock synchronization algorithm based on components originally developed at UCLA to assure that events can be time stamped to a 1 msec jitter. Each vehicle, then maintains an internal estimator of its neighbor's state that runs in a synchronous fashion. In this way, we can force our asynchronous consensus filter to appear as if it were operating in a synchronous manner.

The following picture shows the output of a java GUI that was written to control and monitor the swarm's operation. The java GUI runs on a desktop system that is listening to network traffic using a MICA2 running the GenericBase application. The menu on the bar has several text fields that are used to address and format commands that can be transmitted to the nodes. The application listens into network traffic which consists primarily of messages declaring the robot's current physical and consensus state. The righthand picture is a photograph of the actual robot positions. The lefthand picture shows the corresponding positions in the java GUI. The robots are represented by blue squares with a bar to indicate the front of the vehicle. The consensus state is shown by the small blue dots in the figure. In this case, we see that one of the robot's consensus state has indeed converged to the true center of the swarm.